ResistanceTracker - PLANNING 1

Date: 23/07/2025

Overview

Purpose of this document is to start laying out high level objectives and direction of the ResistanceTracker project. Distilling the idea into tangible concepts and identifying any questions at this stage. End result should be a project outline from which more granular planning can be produced.

General Concept

Resistance Tracker is a browser application for tracking resistance training and results.

The key features include:

* Ability to save workout session data
* Ability to save personal data and measurements
* Ability to interrogate historic data at various levels
* Analysis tools ie; graphs
* Planning tools for sessions and programs
* ‘Journal’ feature for saving training notes
* Upload functions for migrating data

Motivations

The main driver for this project is to learn and practice web development technologies and techniques.

The secondary driver, and the reason for this specific project, is that it is something I will personally use. I already track my training using a spreadsheet and intend to migrate this to ResistanceTracker, which gives me a purpose to complete this project, in effect having a customer to produce it for, which is myself.

Tech Stack

As stated above the main motivation for this project is to learn and practice web development. I have already produced a prototype for this implementing the following technologies:

**Frontend**

HTML / CSS - Display

Javascript - Logic

**Backend**

Node.js – Logic (Use same fundamental language for front and back end, already on a huge learning curve)

Express – Web server framework (abstract away http layer stuff)

Sqlite3 – Database (using sqlite library. Suitable for current single user use and learning database management)

Design guidelines

The following intentions should be followed throughout design and planning stages;

* Functionality > Aesthetics – I am least proficient in frontend design and ‘style’ work, and I don’t care as much about appearance. I want this to work well before it looks good. The aesthetics can be improved later if necessary.
* Consistent style – That being said, all visual design choices should be coherent, and as such, a ‘style guide’ should be established fairly early on. Once the first functional components have been constructed, a period of time should be spent establishing a defined style which can be applied to all future components and which can be evolved as a whole over time.
* Modular Components – Functionality should as far as possible be self contained in such a way that allows for a minimum viable product to be developed relatively quickly, but can be scaled and built upon in future without major redesign.
* Modular Development – Following on from the previous point, discrete components should be fully developed and implemented sequentially, as opposed to gradually building multiple at a time.
* Functional Neutrality/Flexibility – By this I mean avoiding creating something specifically tailored to myself and my current training methods. Firstly because they are liable to change over time, and it is good practice to develop something that caters to a wide range of users. This is especially true in the realm of training as there are so many approaches and tendencies. Even if this product is never scaled to public release, it is good practice to consider this. This will mean flexibility in session and program planning, data tracking, analysis etc..
* Prioritise simplicity – This ties into the above point somewhat. As stated previously, I currently use a spreadsheet to track my training which is both flexible and simple. I want to develop something that maintains these aspects while abstracting away the manual nature of analysis.

Next Steps

**Specifications**

While the immediate aim is to produce a minimally viable product, it is worth looking ahead to likely future features, so that initial features can be designed with scalability in mind

* Clearly define scope
* Detailed definitions of necessary features
* Database structure

**Technical Design**

A replicable plan should be clearly defined for the ‘build’ of the project. This will aid in troubleshooting down the line, as well as potential major redesigns that may occur in early development due to any architectural mistakes – these will be best caught early and dealt with as efficiently as possible to avoid the project unravelling.

* Fundamental infrastructure design
* Development, staging, implementation workflow design

**Project planning**

Plan schedule for early development stages.

**Early Development**

Essentially proof of concept stage, the aim is to test the assumptions and design choices made in the planning and design stage by developing an absolutely minimum product with the essential features present. Once any mistakes have been corrected and the functionality is there, visual design ‘style’ can be established.

* Development of first key features
* Develop the visual style